Lecture 9

Brad McNeney

2017-03-14

Load packages

```r
library(broom)
library(ggplot2)
library(dplyr)
library(gapminder)
data(gapminder)
```

Topics

- Exploratory data analysis, with emphasis on `ggplot2` graphics, using the gapminder data.
    - Suppose we want to use information on `continent`, `year`, `pop` and `gdpPercap` to predict `lifeExp`.
- Base R graphics *vs* grid graphics
- Introduction to `ggplot2`

Exploratory analyses

Exploratory analyses

- Univariate summaries, such as means/medians, sds/IQRs, histrograms, to examine distributions and identify possible measurement errors.
- Pair-wise correlations, to look for relationships between variables
- Pair-wise regression relationships and added-variable-plots
    - Trends over time deserve special attention

Univariate Summaries

- Different summaries are appropriate for categorical and quantitative variables
  - Tabulate categorical variables
  - Five number summary for quantitative variables

```r
summary(gapminder)
```

```
##         country        continent        year         lifeExp
##  Afghanistan:  12   Africa  :624   Min.   :1952   Min.   :23.60
##  Albania    :  12   Americas:300   1st Qu.:1966   1st Qu.:48.20
##  Algeria    :  12   Asia    :396   Median :1980   Median :60.71
##  Angola     :  12   Europe  :360   Mean   :1980   Mean   :59.47
##  Argentina  :  12   Oceania : 24   3rd Qu.:1993   3rd Qu.:70.85
##  Australia  :  12                  Max.   :2007   Max.   :82.60
##  (Other)    :1632
##       pop              gdpPercap
##  Min.   :6.001e+04  Min.   :   241.2
##  1st Qu.:2.794e+06  1st Qu.:  1202.1
##  Median :7.024e+06  Median :  3531.8
##  Mean   :2.960e+07  Mean   :  7215.3
##  3rd Qu.:1.959e+07  3rd Qu.:  9325.5
##  Max.   :1.319e+09  Max.   :113523.1
##
```

Comments on summaries

- Observations in pop and gdpPercap differ by orders of magnitude
  - May be more informative to consider transformations of these variables.
  - For example, a log-10 transformation: one-unit differences correspond to 10-fold increases.
- Aside: Which country has per-capita GDP of $113,523? Or more generally, which observations are in, say, the top 0.1%?

```
filter(gapminder,gdpPercap > quantile(gdpPercap,0.999))
```

```
## # A tibble: 2 × 6
##   country continent  year lifeExp    pop gdpPercap
##    <fctr>    <fctr> <int>   <dbl>  <int>     <dbl>
## 1  Kuwait      Asia  1957  58.033 212846  113523.1
## 2  Kuwait      Asia  1972  67.712 841934  109347.9
```

Unviariate summaries by grouping variable

- May be of interest to do summaries of some variables stratified by a grouping variable.

```
oldops <- options(tibble.width=Inf, tibble.print_max=Inf)
gm_byContinent <- group_by(gapminder,continent)
summarize(gm_byContinent,min(lifeExp),median(lifeExp), IQR(lifeExp),mean(lifeEx
```

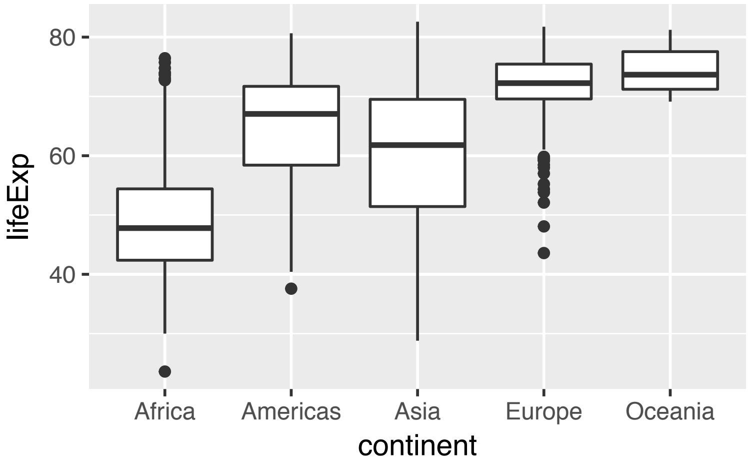
```
## # A tibble: 5 × 7
##   continent `min(lifeExp)` `median(lifeExp)` `IQR(lifeExp)`
##      <fctr>          <dbl>             <dbl>          <dbl>
## 1    Africa         23.599           47.7920        12.0390
## 2  Americas         37.579           67.0480        13.2895
## 3      Asia         28.801           61.7915        18.0790
## 4    Europe         43.585           72.2410         5.8805
## 5   Oceania         69.120           73.6650         6.3475
##   `mean(lifeExp)` `sd(lifeExp)` `max(lifeExp)`
##             <dbl>         <dbl>          <dbl>
## 1        48.86533      9.150210         76.442
## 2        64.65874      9.345088         80.653
## 3        60.06490     11.864532         82.603
## 4        71.90369      5.433178         81.757
## 5        74.32621      3.795611         81.235
```

```
options(oldops)
```

Boxplots

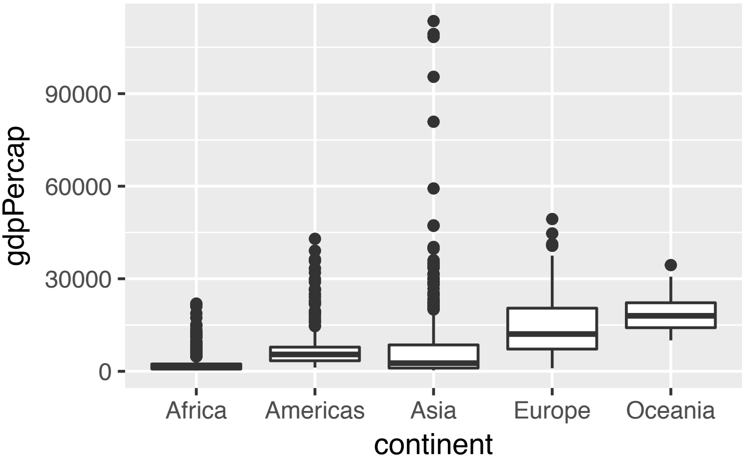
- Graphical representation of the five number summary for grouped data

```
ggplot(gapminder,aes(x=continent, y=lifeExp)) + geom_boxplot()
```



Boxplots, cont.

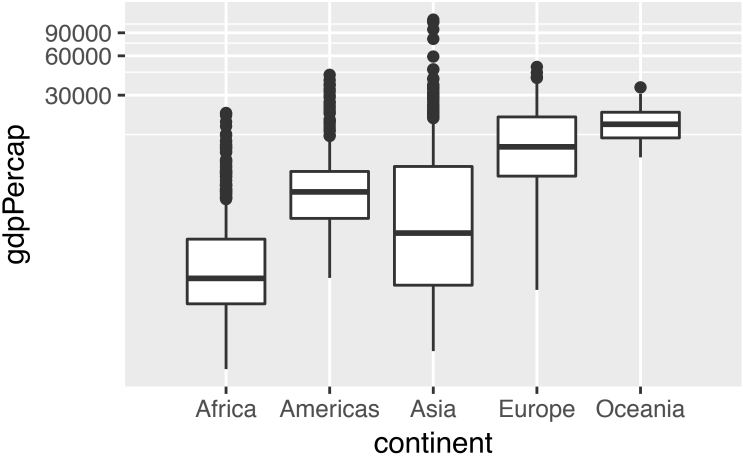
```
ggplot(gapminder,aes(x=continent, y=gdpPercap)) + geom_boxplot()
```



- Distribution of log-transformed data may be more informative.

Boxplots, cont.

```
ggplot(gapminder,aes(x=continent, y=gdpPercap)) +
  coord_trans(y="log10") + geom_boxplot()
```



Adding transformed variables to a dataset

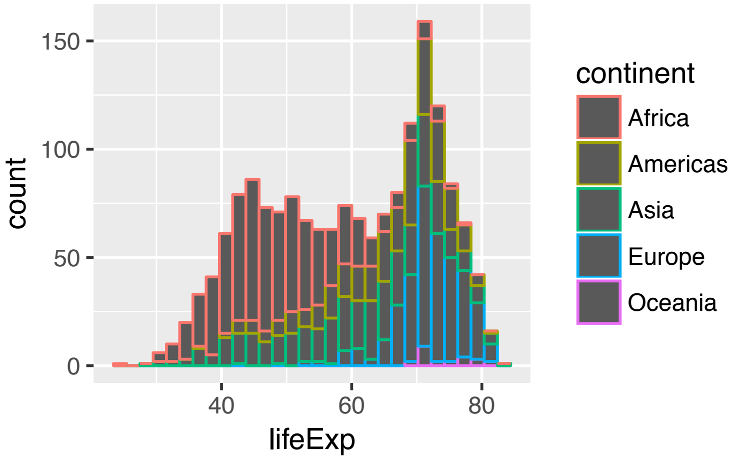
- Above suggests we add log of `gdpPercap` to the dataset.
- A similar exploration of the `pop` variable suggests we include log of `pop` too.
- Will use log-base-10.

```
gapminder <- mutate(gapminder,
                    log10Pop = log10(pop),
                    log10GdpPercap = log10(gdpPercap))
```

Histograms

- Shows the shape of distributions and can suggest possible outliers
- Stacked histograms:

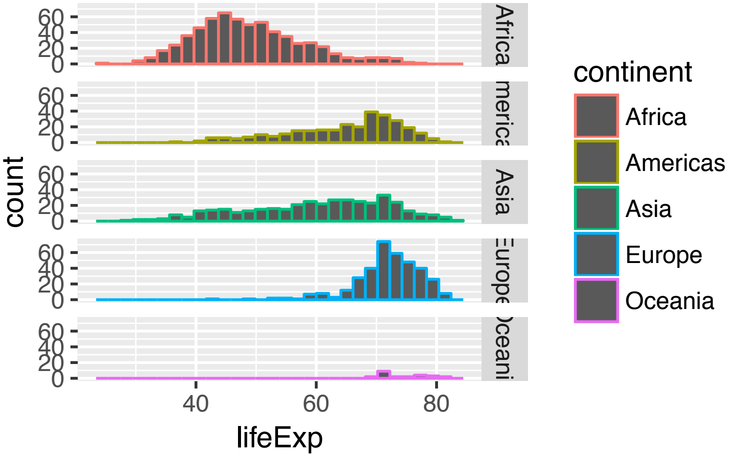
```
ggplot(gapminder,aes(x=lifeExp, color=continent)) + geom_histogram()
```



Histograms, continued

- Histograms in different plot panels, or “facets”:

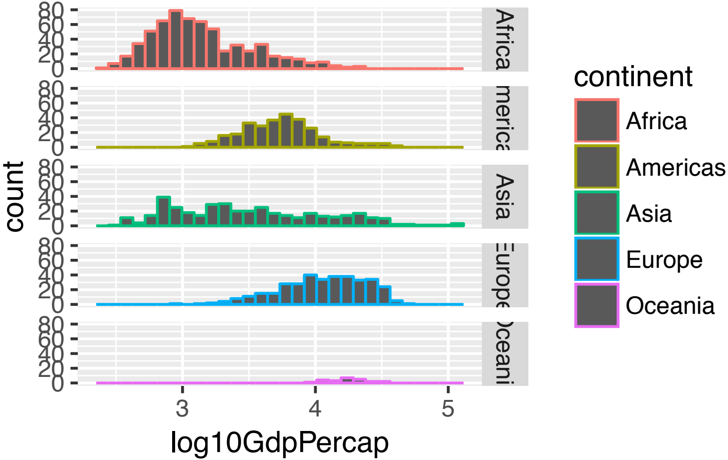
```
ggplot(gapminder,aes(x=lifeExp, color=continent)) +
  geom_histogram() + facet_grid(continent ~ .)
```



Histograms of the explanatory variables

- May also be of interest

```
ggplot(gapminder,aes(x=log10GdpPercap, color=continent)) +
  geom_histogram() + facet_grid(continent ~ .)
```

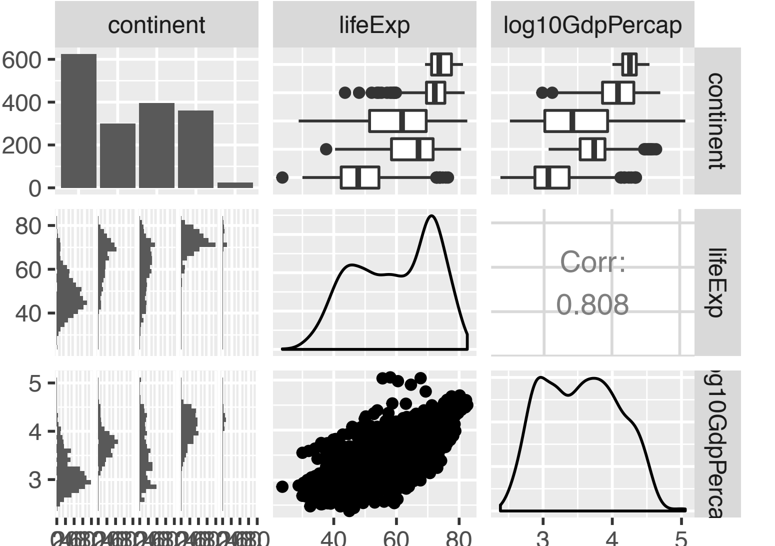


Pairwise Regression relationships

- Though pairwise relationships don't tell the whole story, they are a useful starting point.
- The `GGally` package provides the function `ggpairs()` to facilitate this.
  - Can do all possible pairs of variables, but I find this too hard to read for more than three variables.

Pairwise plots

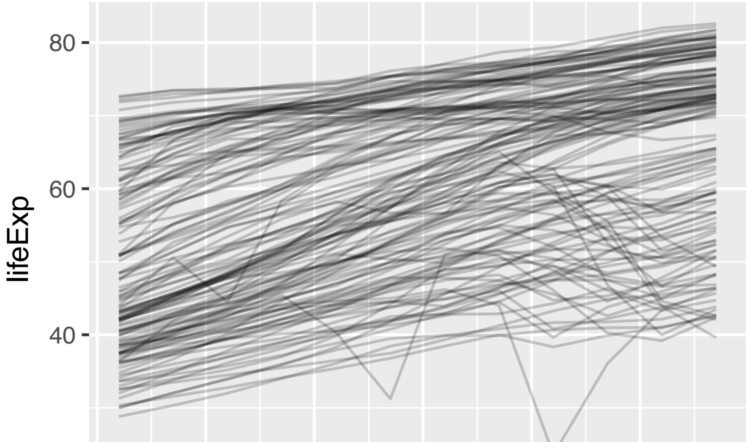
```
library(GGally)
gm_sub <- select(gapminder,continent,lifeExp,log10GdpPercap)
ggpairs(gm_sub) # Cut and paste into console to see better
```



Time trends

- Can represent time series by lines.
- There are many time series in these data – need to make each line slightly transparent to account for overplotting

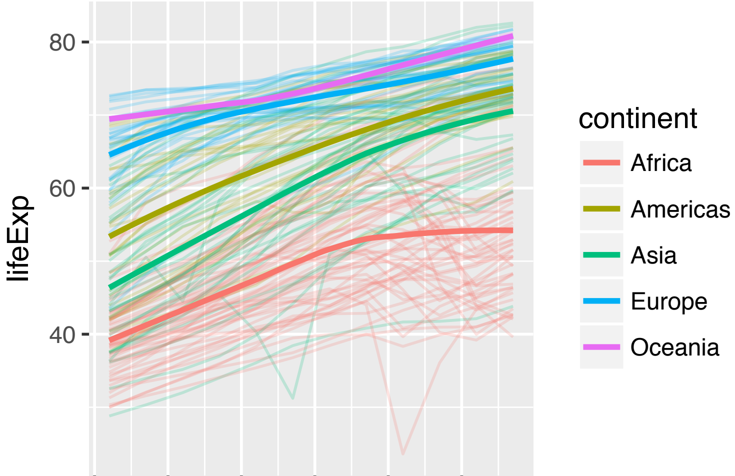
```
ggplot(gapminder,aes(x=year,y=lifeExp,group=country)) +
  geom_line(alpha=0.2)
```



Time trends, cont.

- Can add a statistical summary, like medians at each time, or a smoother.
- Can also add colours for different continents.

```
ggplot(gapminder,aes(x=year,y=lifeExp,group=country,
                     color=continent)) + geom_line(alpha=0.2) +
  geom_smooth(aes(x=year,y=lifeExp,group=continent),se=FALSE)
```



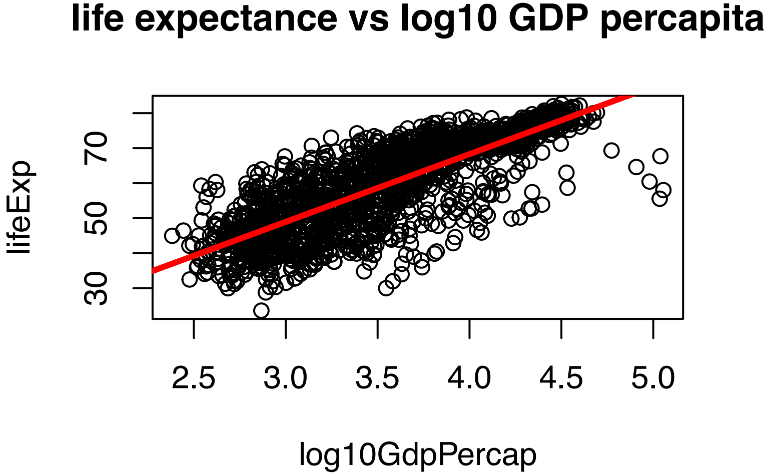
Base R graphics

Base R graphics

- Very serviceable graphics system capable of producing publication-quality graphs.
- Create graphics by calling functions that either produce complete plots or add to plots
- Like adding paint to a canvas

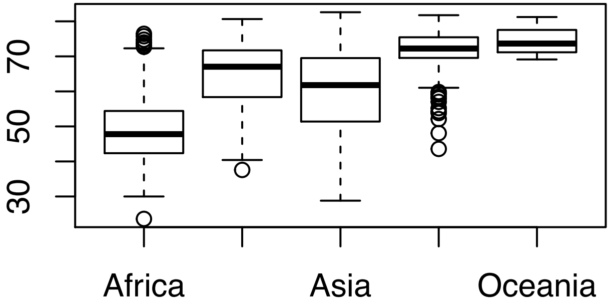
Base R examples

```r
with(gapminder,plot(log10GdpPercap,lifeExp)) # or plot(lifeExp ~ log10GdpPercap
title(main="life expectance vs log10 GDP percapita")
abline(lm(lifeExp ~ log10GdpPercap,data=gapminder),col="red",lwd=3)
```



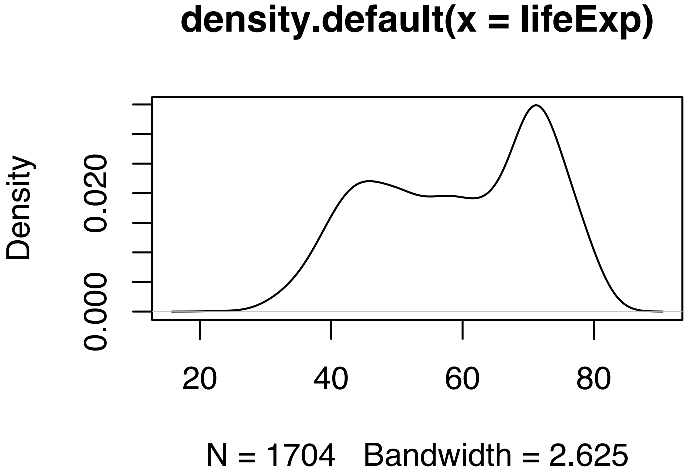
Base R examples

```
with(gapminder,boxplot(split(lifeExp,continent)))
```



Base R examples

```r
with(gapminder,plot(density(lifeExp)))
```



Base R graphics: Where to learn more

- Paul Murrell's book: [`https://www.stat.auckland.ac.nz/~paul/RG2e/`]
- Ross Ihaka's lectures: [`https://www.stat.auckland.ac.nz/~ihaka/787/lectures-r-graphics.pdf`]

`grid` graphics

`grid` graphics

- `grid` graphics is a low-level graphics system that allows fine control of graphics elements
- Users can create multiple graphics regions, or “viewports”, that are arranged on the graphics device or nested within one another.
- Graphical objects, or “grobs” can be created and drawn on these viewports (e.g., lines, shapes).
- Grobs can be editted (e.g., change color of lines) and re-drawn

grid graphics: Where to learn more

- Paul Murrell's book: [https://www.stat.auckland.ac.nz/~paul/RG2e/]

ggplot2 basics

`ggplot2`

- `ggplot2` is implemented in `grid` graphics
- The g's stand for Grammar of Graphics.
  - Like English grammar is the way in which words are put together to form sentences, a grammar of graphics is a way to put together basic graphical elements to make a graph.
- To understand the grammar we need to define the basic elements.
  - Start with definitions (in bold), some of which are too abstract to be useful until we get into details.
- `ggplots` can be built in layers, comprised of **data** a **mapping**, a **geom** and optionally **stats**
- The layers are arranged and labelled on the graph by **scale**s and **coord**s.
- The data can also be broken into subsets and displayed in separate graphs by a **facet** specification.

Components of a ggplot: data and mappings

- We start with the **data** we want to visualize and a **mapping**, or aesthetic, that describes how these data map to attributes on the plot.

```
p <- ggplot(gapminder,aes(x=log10GdpPercap,y=lifeExp,color=continent))
```

- From the dataset gapminder, the variable log10GdpPercap will be mapped to y-coordinates, lifeExp will be mapped to the x-coordinates, and continent will be perceived as colours.

Components of a `ggplot`: geometric objects (geoms)

- Geometric objects or **geoms** are things like points and lines that we see on the plot.

```
p2 <- p + geom_point(alpha=0.5)
```

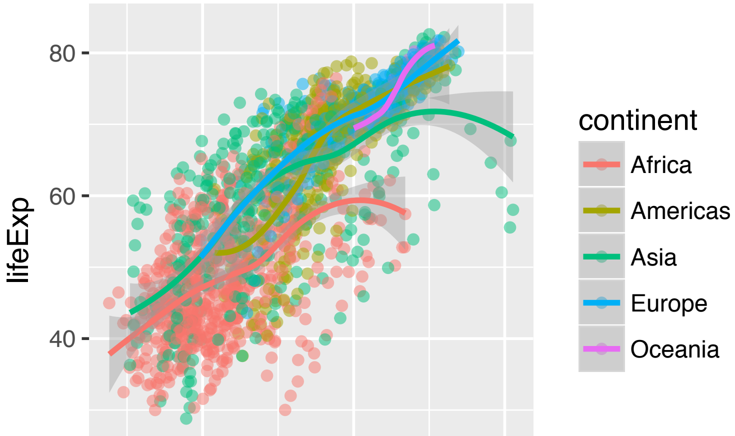
- alpha is the transparency aesthetic, between 0 and 1, best applied directly to the geom it is to apply t

Components of a ggplot: statistical transformations

- Statistical transformations or **stats** summarize the data; e.g., a scatterplot smoother

```
p2 + geom_smooth()
```

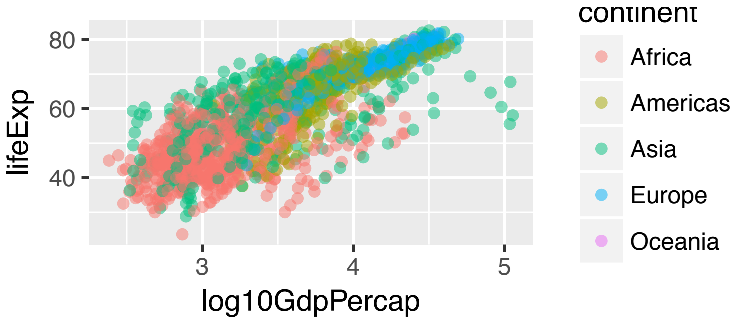
```
## `geom_smooth()` using method = 'loess'
```



Components of a `ggplot`: scales

- The **scales** are mappings from the data to the graphics device
  - domain of `continent` is the five continents, range is the hexidecimal of the five colors represented on the graph
  - domain of `lifeExp` is 23.599 to 82.603, range is [0,1], which `grid` converts to a range of vertical pixels on the graph.
  - legends and axes provide the inverse mapping

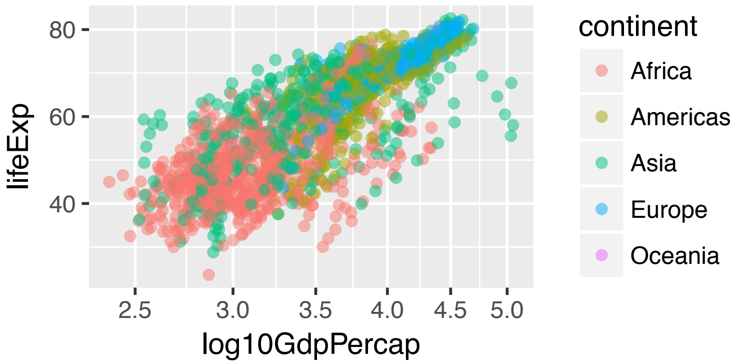
```
p2
```



Components of a ggplot: coodinate system

- The coordinate system is another layer in how the data get mapped to the graphics device.
  - Usually Cartesian, but could be, e.g., polar coordinates, or a transformation.

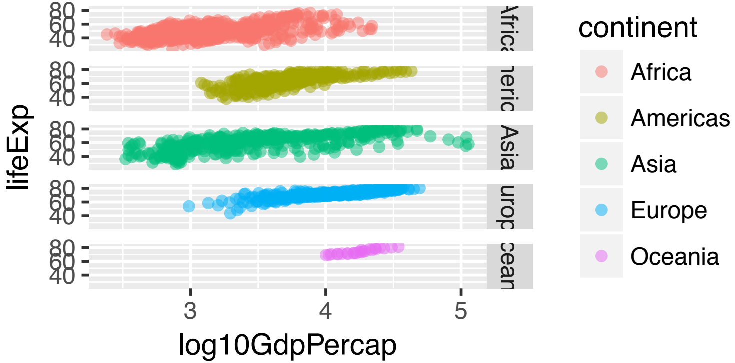
```
p2 + coord_trans(x="log10")
```



Components of a ggplot: faceting

- How to break up the data into subsets and arrange multiple plots on the graphics device.

```r
p2 +  facet_grid(continent ~ .)
```



Why so many components?

- A framework for the components of a graph.
- Gives the user the ability to change indvidual components one at a time.

More details

- Layers
  - data, aesthetic mapping, geom, statistical transformation and position adjustment (to be defined)
- Tools for working with layers
- Scales, axes and legends
- Positioning: faceting and coordinate systems

Example dataset: Diamonds

- Price and quality of about 54,000 diamonds.
- Quality measures are carat (size), cut, colour and clarity
- Also included are three measures of the dimensions of each diamond labelled x, y and z.

```r
data(diamonds)
head(diamonds)
```

```
## # A tibble: 6 × 10
##   carat       cut color clarity depth table price     x     y     z
##   <dbl>     <ord> <ord>   <ord> <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  0.23     Ideal     E     SI2  61.5    55   326  3.95  3.98  2.43
## 2  0.21   Premium     E     SI1  59.8    61   326  3.89  3.84  2.31
## 3  0.23      Good     E     VS1  56.9    65   327  4.05  4.07  2.31
## 4  0.29   Premium     I     VS2  62.4    58   334  4.20  4.23  2.63
## 5  0.31      Good     J     SI2  63.3    58   335  4.34  4.35  2.75
## 6  0.24 Very Good     J    VVS2  62.8    57   336  3.94  3.96  2.48
```

Layers

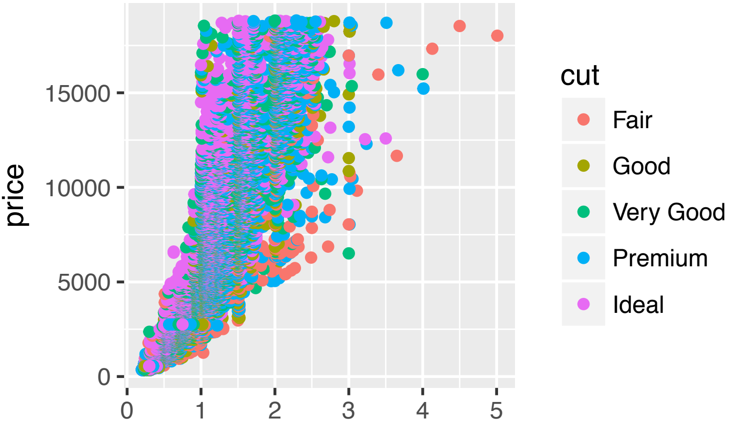
- We first initialize the plot and then start adding layers.
- Initializing is done with `ggplot()`.
  - We usually specify the default data and aesthetic mappings for all subsequent layers, though this is not necessary.
  - Without layers, the plot is not displayed.

```r
p <- ggplot(diamonds,aes(x=carat,y=price,colour=cut))
```

Adding layers

- Add with a +
- The `layer()` function can be used to specify a geom, stat and position
    - data and mapping will be inherited from initialization

```
p + layer(geom="point", stat="identity", position="identity")
```



Shortcuts

- Shortcut functions are of the form `geom_XXX()` and `stat_XXX()`.
  - The `geom_XXX()` functions have a default stat and position
  - The `stat_XXX()` functions have a default geom and position
  - The `geom_XXX()` can over-ride the default stat and the `stat_XXX()` can over-ride the default geom though
- Call on the previous slide is equivalent to

```
p <- p + geom_point()
```

Plot objects

- Notice that plot objects can be stored as R objects:

```
summary(p)
```

```
## data: carat, cut, color, clarity, depth, table, price, x, y, z
##   [53940x10]
## mapping:  x = carat, y = price, colour = cut
## faceting: <ggproto object: Class FacetNull, Facet>
##     compute_layout: function
##     draw_back: function
##     draw_front: function
##     draw_labels: function
##     draw_panels: function
##     finish_data: function
##     init_scales: function
##     map: function
##     map_data: function
##     params: list
##     render_back: function
##     render_front: function
##     render_panels: function
##     setup_data: function
##     setup_params: function
##     shrink: TRUE
##     train: function
##     train_positions: function
##     train_scales: function
##     vars: function
##     super:  <ggproto object: Class FacetNull, Facet>
## -----------------------------------
## geom_point: na.rm = FALSE
## stat_identity: na.rm = FALSE
## position_identity
```